Gesture Recognition Assignment – Using CNN Models

Data Scientist: Shashank Pawaskar

Problem Statement

Imagine you are working as a data scientist at a home electronics company which manufactures state of the art smart televisions. You want to develop a cool feature in the smart-TV that can recognise five different gestures performed by the user which will help users control the TV without using a remote.

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

Gesture	**Corresponding Action**
Thumbs Up	Increase the volume.
Thumbs Down	Decrease the volume.
Left Swipe	'Jump' backwards 10 seconds.
Right Swipe	'Jump' forward 10 seconds.
Stop	Pause the movie.

Each video is a sequence of 30 frames (or images).

Goal of the Assignment

- **Generator**: The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization should be performed successfully.

- **Model**: Develop a model that is able to train without any errors which will be judged on the total number of parameters (as the inference(prediction) time should be less) and the accuracy achieved.
- **Write up**: This should contain the detailed procedure followed in choosing the final model. The write up should start with the reason for choosing the base model, then highlight the reasons and metrics taken into consideration to modify and experiment to arrive at the final model.

This document is to fulfill the 3rd Goal of the Assignment, a document which explains the process employed to derive at the Final Model.

Journey of Model Development

1. **Setup and Data Preparation**
   - 1.1. First Step was to identify the Libraries required, and as I progressed, added few more libraries during the course of Model Development.
   - 1.2. **Customer Image Data Generator / Augmentor** : Developed it to using the Sample Code provided, but had to refer to sites and the Code Development during the course
     - 1.2.1. https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/
     - 1.2.2. https://github.com/keras-team/keras-preprocessing/blob/master/keras_preprocessing/image/image_data_generator.py
   - 1.3. **Identification of Right Parameters for the Model**
     - 1.3.1. Tried to use Keras Tuner, but could not due the known bug/issue
       - 1.3.1.1. https://github.com/keras-team/keras/issues/17368
     - 1.3.2. Tried to create a "For" loop to run a model with different Frame Sizes, Batch Sizes, Kernel / Pool Size, Dropout, Dense Neurons, Bias….so on.
       - 1.3.2.1. Dropped this idea, as it was getting difficult to keep track of the results and required bit more coding than I expected, which was not possible to complete in the given time frame for the Assignment.
   - 1.4. Class for Functions referred - https://pynative.com/python-class-method/
   - 1.5. Class for Model referred - https://www.javatpoint.com/keras-the-model-class
2. **Goals / Objective of the Assignment**
   - 2.1. Built Generator and Tested it for both Train and Validation Image Data Sets
     - 2.1.1. Included Data Augmentation
     - 2.1.2. Parameters to a large extent, which helped in avoiding repeated set of code lines during the Model Development.
   - 2.2. Build Models based on the initial Experiments for Conv3D.
   - 2.3. Built Models with Conv2D + RNN (Variants LSTM, GRU, ComvLSTM)
   - 2.4. Build Model using Transfer Learning (MobilNet), did not use v2 and V3 of MobilNet.
     - 2.4.1. Wanted to use VGG16, but due to time constraints, restrained from using it.

Next Section provides the detail of each model developed and run.

1. Initiated a set of experiments for Conv 3D models to run for 3 Epochs, to arrive at the parameters.
2. Set 1 Experiments for Conv 3D models were customized layered experiments, the inference derived was they could be improved with changes in layers and other parameters.
    a. Key Observations :
        i. The Model when executed, ran into GPU Memory error, this was Batch Size of 30.
        ii. Also, the early indications that the Set 1 Experiment Models may Under-Fit or Over-Fit.
        iii. Decision Taken : Not to Pursue with this model architecture.
3. Set 2 Experiments for Conv 3D models were based on one single Model Architecture and tuning of the parameters.
    a. Key Observations:
        i. Applied Learning Set 1 Experiments and used Batch Size of 16, while experimenting the other parameters like image dimensions, Frames, Drop Out and Dense(Hidden Layer) Neurons

Model Development was taken up both sets of experiments and following are the results documented:

Experiment Number #	Model	Result	Decision + Explanation
1	CNN3D Set 2 Model Architecture : Model 1	**Data Augment = True** Model Name : Set2_Model_1_History # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (100 , 100) # No. of Epochs : 25 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 898,325 # Trainable Params : 897,973 # Non Trainable Params : 352 # Max. Training Accuracy : 0.942 # Max. Validation Accuracy : 0.821	Better Training and Validation Loss and Accuracy Loss Training and Validation Loss and Accuracy trending nicely, but with some peaks and valleys ****model-00022-0.19102-0.94048-0.33544-0.75893.h5**** - with Training Loss-0.1910, Validation Loss - 0.3354, Training Accuracy - 0.9405, Validation Accuracy - 0.7589, are the best rates for the model. **Decision :** To check whether if we can Further Improve Loss and Accuracies
2	CNN3D Set 2 Model Architecture : Model 2	**Data Augmentation = False** Model Name : Set2_Model_2_History # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (100 , 100)	Loss Training and Validation Loss and Accuracy trending nicely, but with less number of peaks and valleys ****model-00024-0.24873-0.92113-0.39267-0.91964.h5****

Experiment Number #	Model	Result	Decision + Explanation
		# No. of Epochs : 25 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 898,325 # Trainable Params : 897,973 # Non Trainable Params : 352 # Max. Training Accuracy : 0.9271 # Max. Validation Accuracy : 0.9196	- with Training Loss- 0.2487, Validation Loss - 0.3927, Training Accuracy - 0.9271, Validation Accuracy - 0.91964, are the best rates for the model. **Decision :** To check whether if we can Further Improve Loss and Accuracies
3	CNN3D Set 2 Model Architecture : Model 3	**Data Augment = True** **Model Name : Set2_Model_3_History** # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (120 , 120) # No. of Epochs : 30 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 1,092,885 # Trainable Params : 1,092,533 # Non Trainable Params : 352 # Max. Training Accuracy : 0.9040 # Max. Validation Accuracy : 0.7143	Callback Early Stopping at Epoch 18. Drop in Training and Validation Loss and Accuracy, than what observed in Model 1 and 2 Loss Training and Validation Loss and Accuracy trending indicate model is bit overfitting Also, the number Training Parameters have increased. ****model-00018-0.28052-0.90402-0.68291-0.63393.h5**** - with Training Loss- 0.2805, Validation Loss - 0.6829, Training Accuracy - 0.9040, Validation Accuracy - 0.6339, are the best rates for the model. **Decision:** To check whether if we can Further Improve Loss and Accuracies, from what we have seen in Experiment 2.
4	CNN3D Set 2 Model Architecture : Model 4	**Data Augment = False** Model Name : Set2_Model_4_History # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (120 , 120) # No. of Epochs : 30 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 1,092,885	Callback Early Stopping at Epoch 7. Drop in Training and Validation Loss and Accuracy, than what observed in Model 1 and 2 Loss Training and Validation Loss and Accuracy trending indicate model is overfitting Also, the training parameter have increased, when compared that for Model 1 and Model 2.

Experiment Number #	Model	Result	Decision + Explanation
		# Trainable Params : 1,092,533 # Non Trainable Params : 352 # Max. Training Accuracy : 0.6711 # Max. Validation Accuracy : 0.3571	**Decision:** To check whether if we can Further Improve Loss and Accuracies, from what we have seen in Experiment 2.
	CNN3D Set 2 Model Architecture : Model 5	**Data Augment = False** **Model Name : Set2_Model_5_History** # No. of Frames : 30 # Batch Size : 30 # Image Dimensions : (120 , 120) # No. of Epochs : 30 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 2,118,821 # Trainable Params : 2,118,373 # Non Trainable Params : 448 # Max. Training Accuracy : 0.8116 # Max. Validation Accuracy : 0.3333	Callback Early Stopping at Epoch 8. Drop in Training and Validation Loss and Accuracy, than what observed in Model 1 and 2 Loss Training and Validation Loss and Accuracy trending indicate model is overfitting Also, the trainable Parameters are considerably higher than that for Model 1 and 2. **Decision:** To check whether if we can Further Improve Loss and Accuracies, from what we have seen in Experiment 2.
	CNN3D Set 2 Model Architecture : Model 6	**Data Augment = True** **Model Name : Set2_Model_6_History** # No. of Frames : 30 # Batch Size : 30 # Image Dimensions : (120 , 120) # No. of Epochs : 30 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 2,118,821 # Trainable Params : 2,118,373 # Non Trainable Params : 448 # Max. Training Accuracy : 0.8116	Encountered Error - "2023-08-07 02:36:41.255741: W tensorflow/core/common_runtime/bfc_allocator.cc:463] Allocator (GPU_0_bfc) ran out of memory trying to allocate 3.09GiB (rounded to 3317760000)requested by op gradient_tape/sequential_1/max_pooling3d_3/MaxPool3D/MaxPool3DGrad If the cause is memory fragmentation maybe the environment variable 'TF_GPU_ALLOCATOR=cuda_malloc_async' will improve the situation.". Decision: To change the parameters and re-run the model.

Experiment Number #	Model	Result	Decision + Explanation
		# Max. Validation Accuracy : 0.3333	
	Re-run with new parameters CNN3D Set 2 Model Architecture : Model 6	**Data Augment = True** **Model Name : Set2_Model_6_History** # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (100 , 100) # No. of Epochs : 30 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 1,729,701 # Trainable Params :1,729,253 # Non Trainable Params : 448 # Max. Training Accuracy : 0.7024 # Max. Validation Accuracy : 0.4196	Callback Early Stopping at Epoch 6. Drop in Training and Validation Loss and Accuracy, than what observed in Model 1 and 2 Loss Training and Validation Loss and Accuracy trending indicate model is overfitting Also, the number Training Parameters have increased (though lower than the Model 5). **Decision:** To check whether if we can Further Improve Loss and Accuracies, from what we have seen in Experiment 2.
7	CNN3D Set 2 Model Architecture : Model 7	**Data Augment = False** **Model Name : Set2_Model_7_History** # No. of Frames : 30 # Batch Size : 30 # Image Dimensions : (100 , 100) # No. of Epochs : 25 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 1,729,701 # Trainable Params : 1,729,253 # Non Trainable Params : 448 # Max. Training Accuracy : 0.7768 # Max. Validation Accuracy : 0.5667	Callback Early Stopping at Epoch 19. Drop in Training and Validation Loss and Accuracy, than what observed in Model 1 and 2 Loss Training and Validation Loss and Accuracy trending indicate model is overfitting Also, the number Training Parameters higher than that for Model 1 and Model 2. **Decision:** To check whether if we can Further Improve Loss and Accuracies, from what we have seen in Experiment 2.
8	CNN3D	**Data Augment = False** **Model Name : Set2_Model_8_History**	Callback Early Stopping at Epoch 7.

Experiment Number #	Model	Result	Decision + Explanation
	Set 2 Model Architecture : Model 8	# No. of Frames : 30 # Batch Size : 30 # Image Dimensions : (100 , 100) # No. of Epochs : 25 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 1,729,701 # Trainable Params : 1,729,253 # Non Trainable Params : 448 # Max. Training Accuracy : 0.4797 # Max. Validation Accuracy : 0.3833	Training and Validation Loss Plot indicates model tending to over-fit. show and Accuracy, trending better but the training parameters are much higher than the Model1 and Model 2 **Decision:** To check whether if we can Further Improve Loss and Accuracies, from what we have seen in Experiment 2.
9	CNN3D Set 2 Model Architecture : Model 9	**Data Augment = False** **Model Name : Set2_Model_8_History** # No. of Frames : 30	Training and Validation Loss Plot indicates model tending to over-fit. show and Accuracy, trending better but the training parameters are much higher than the Model1 and Model 2 **Decision:** To check whether if we can Further Improve Loss and Accuracies, from what we have seen in Experiment 2.
10	CNN3D Set 2 Model Architecture : Model 10	**Data Augment = False** **Model Name : Set2_Model_10_History** # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (100 , 100) # No. of Epochs : 30 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 108,117 # Trainable Params : 107,637 # Non Trainable Params : 480 # Max. Training Accuracy : 0.5536 # Max. Validaiton Accuracy : 0.3393	Callback Early Stopping at Epoch 8. Training and Validation Loss and Accuracy trend indicates model tending to Over-Fit. **Decision:** To check whether if we can Further Improve Loss and Accuracies, from what we have seen in Experiment 2.

Experiment Number #	Model	Result	Decision + Explanation
11	CNN3D Set 2 Model Architecture : Model 11	**Data Augment = True** **Model Name : Set2_Model_8_History** # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (100 , 100) # No. of Epochs : 40 # Conv3D Layer Kernel Size : (1, 3, 3) # MaxPool3D Pool Size : (1, 2, 2) # Total Params : 108,117 # Trainable Params : 107,637 # Non Trainable Params : 480 # Max. Training Accuracy : 0.5268 # Max. Validation Accuracy : 0.3304	Result similar to Model 10. Stopping the creating more models with Conv 3D. CNN3D Set 2 Model Architecture: Model 2 has the best Loss and Accuracy Score.
CNN3D Set 2 Model Architecture: Model 2 has the best Loss and Accuracy Score. **With Training Loss- 0.2487, Validation Loss - 0.3927, Training Accuracy - 0.9271, Validation Accuracy - 0.91964, are the best rates for the model.** **H5 File : "model-00024-0.24873-0.92113-0.39267-0.91964.h5"**			
Moving on to Conv2D with RNN Model Results			
1	Conv 2D + LSTM : Model 1	**Data Augment = False** **Model Name : CNN2D_Model_1_History** # No. of Frames : 30 # Batch Size : 30 # Image Dimensions : (100 , 100) # No. of Epochs : 40 # Conv3D Layer Kernel Size : (3, 3) # MaxPool3D Pool Size : (2, 2) # Total Params : 698,789 # Trainable Params : 698,277 # Non Trainable Params : 512	Model Training and Validation Plots for Loss and Training are trending in right direction. Training Parameters are lower than the ones for the Conv3D Model 2 (so far the best model). Decision: Increase the number of Epochs and check for better results.

Experiment Number #	Model	Result	Decision + Explanation
		# Max. Training Accuracy : 0.76087 # Max. Validation Accuracy : 0.5917	
2	Conv 2D + LSTM : Model 2	**Data Augment = False** **Model Name : CNN2D_Model_1_History** # No. of Frames : 30 # Batch Size : 30 # Image Dimensions : (100 , 100) # No. of Epochs : 40 # Conv3D Layer Kernel Size : (3, 3) # MaxPool3D Pool Size : (2, 2) # Total Params : 698,789 # Trainable Params : 698,277 # Non Trainable Params : 512 # Max. Training Accuracy : 0.76087 # Max. Validation Accuracy : 0.5917	
3	Conv 2D + GRU	**Data Augment = False** **Model Name : CNN2D_Model_3_History** # No. of Frames : 30 # Batch Size : 20 # Image Dimensions : (100 , 100) # No. of Epochs : 40 # Conv3D Layer Kernel Size : (3, 3) # MaxPool3D Pool Size : (2, 2) # Total Params : 114,405 # Trainable Params : 114,405 # Non Trainable Params : 640 # Max. Training Accuracy : 0.7176 # Max. Validation Accuracy : 0.3899	

Experiment Number #	Model	Result	Decision + Explanation
4	CNN 2D + Transfer Learning with GRU	**Data Augment = False** **Model Name : CNN2D_Model_4_History** # No. of Frames : 30 # Batch Size : 20 # Image Dimensions : (100 , 100) # No. of Epochs : 25 # Conv3D Layer Kernel Size : (3, 3) # MaxPool3D Pool Size : (2, 2) # Total Params : 3,446,725 # Trainable Params : 3,422,789 # Non Trainable Params : 23,936 # Max. Training Accuracy : 0.9750 # Max. Validation Accuracy : 0.9499	Transfer Learning using "MobileNet" Callback – Early Stopping at Epoch 12 Better Training and Validation Loss and Accuracy Loss Training and Validation Loss trend indicate some overfitting. **** model-00010-0.04013-0.96912-0.25982-0.94000.h5**** - With Training Loss-0. 0401, Validation Loss - 0.2598 and Training Accuracy - 0.9691, Validation Accuracy - 0.9400, are the best rates for the model. These results are better than the Conv3D Model 2, but the Training Parameter Count is high, which is expected as we are using Transfer Learning. **Decision :** To check whether if we can Further Improve Loss and Accuracies with lower Training Parameters
5	CNN2D + CnvLSTM2D + GolbalAveragePool	**Data Augment = True** **Model Name : CNN2D_Model_5_History** # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (100 , 100) # No. of Epochs : 50 # Total Params : 22,181 # Trainable Params : 21,941 # Non Trainable Params : 240 # Max. Training Accuracy : 0.8557 # Max. Validation Accuracy : 0.9286	Validation Accuracy Trends Higher than Training Accuracy, also the Validation Loss trends lower than Training Loss. Is it due Data Augmentation, though this helps with Overfitting problem, but is it impacting the model such that Training Accuracy is lower than Validation Accuracy and vice-a-versa in case of Loss. Lets find out by switching of the Data Augmentation for the above model and run the model https://keras.io/getting_started/faq/#why-is-my-training-loss-much-higher-than-my-testing-loss

Experiment Number #	Model	Result	Decision + Explanation
6	CNN2D + CnvLSTM2D + GolbalAveragePool	**Data Augment = False** **Model Name : CNN2D_Model_6_History** # No. of Frames : 30 # Batch Size : 16 # Image Dimensions : (100 , 100) # No. of Epochs : 50 # Total Params : 12,725 # Trainable Params : 12,581 # Non Trainable Params : 144 # Max. Training Accuracy : 0.8928571343421936 # Max. Validation Accuracy : 0.9107142686843872	Removing Data Augmentation shows some change, but the Validation Loss and Accuracy trending better than the Training Loss and Accuracy. Also, the Validation Data set size (in terms of number of videos with respective frames), this may be possible. ConvLSTM2D model performances (trainings/validations/predictions) are encouraging and calls for improvement through a hyper parameter tuning process based on allocation of higher dense-layer filters and lower kernel sizes, especially keeping the Validation Dataset in mind. Due to limited time, concluded the Model Trainings here (To meeting assignment Timeline). **** model-00039-0.33804-0.87351-0.34433-0.91071.h5**** - With Training Loss-0. 3380, Validation Loss - 0.3443 and Training Accuracy - 0.8735, Validation Accuracy - 0.9107, are the best rates for the model. Also, the Training Parameters are way lower than the for Conv3D Model 2.
Final Model Conv2D + Conv2DLSTM2D : Model 6 should perform better. **With Training Loss-0. 3380, Validation Loss - 0.3443 and Training Accuracy - 0.8735, Validation Accuracy - 0.9107, are the best rates for the model.** **H5 File : model-00039-0.33804-0.87351-0.34433-0.91071.h5**			
	Final Model	**Conv2D + Conv2DLSTM2D** **H5 File :** **model-00039-0.33804-0.87351-0.34433-0.91071.h5**	**Decision: Conv2D+Conv2dLSTM2D can be chosen, as the results are fairly decent, it is light in terms of the number Parameters to Train, which are considerably lower than the Conv3D Final Model 2.**

Summary: As per the understanding provided (during the Online **ML C48 Case Study Pre-Assignment Session** on 30th July 2023) Gesture Recognition Assignment was to build a model either with Conv3D or Conv2D + RNN. I have developed the models using both and following are the Final Models (recommended based on the results):

1. For Conv3D
    a. CNN3D Set 2 Model Architecture: Model 2 has the best Loss and Accuracy Score.
        i. H5 File : "**model-00024-0.24873-0.92113-0.39267-0.91964.h5**"
2. For Conv2D + RNN : Final Model Chosen, as it light weight in terms of number of Parameters to Train and also has decent Loss and Accuracy outcome.
    a. Conv2D + Conv2DLSTM2D : Model 6
        i. H5 File : "**model-00039-0.33804-0.87351-0.34433-0.91071.h5"**